2. Syntax von Programmiersprachen

Die Beschreibung von Programmiersprachen gliedert sich in Syntax und Semantik. Allgemein wird gesagt, daß die Syntax das textuelle Erscheinungsbild des Programms definiert, während die Semantik seine Bedeutung festlegt. Eine Sprache mit formal definierter Syntax heißt formale Sprache. Insbesondere ist die Syntax einer Programmiersprache eine formale Sprache. In diesem Kapitel betrachten wir zwei Beschreibungsformalismen für die Syntax von Programmiersprachen: Syntaxdiagramme und EBNF (Erweiterte Backus-Naur-Form).

Beispiel für eine Grammatik:

als Syntaxdiagramm: Satz: →Subjekt→Prädikat→Objekt→

in EBNF-Form: Satz = Subjekt Prädikat Objekt

2.1 Syntaxdiagramme

Ein **Syntaxdiagramm** ist ein einfacher grafischer Formalismus zur Beschreibung der Syntax von Programmen.
Ein **Syntaxdiagramm** enthält 2 Arten von Symbolen:

Nichtterminalsymbole (umrahmt durch Rechtecke) und

Terminalsymbole (umrahmt durch Ovale).

Dabei bezeichnen die Terminalsymbole die Grundelemente der formalen Sprache („die Buchstaben des Alphabets“), während die Nichtterminalsymbole Begriffe bezeichnen, die durch das Syntaxdiagramm erklärt werden.
Gegeben je eine Menge von Terminal- und Nichtterminalsymbolen besteht ein Syntaxdiagramm aus einer Menge von Regeln der Form:

Nichtterminalsymbol:

Diagramm

wobei ein Diagramm aus Terminal- und Nichtterminalsymbolen besteht, die durch Pfeile miteinander verbunden sind. Jedes Diagramm hat genau einen Eingang und genau einen Ausgang.

Beispiel:

Satz:

2.2 Erweiterte Backus-Naur-Form EBNF

Die **E**rweiterte **B**ackus-**N**aur-**F**orm EBNF, benannt nach John Backus und Peter Naur, wurde erstmals zur Beschreibung der Syntax von Algol 60 verwendet. Heute ist die BNF (in notationellen Varianten) die Standardbeschreibungstechnik für Programmiersprachen. Auch die Syntax von Java wird in dieser Form beschrieben.

.

Wie bei Syntaxdiagrammen unterscheiden wir 2 Arten von Symbolen:

- **Nichtterminalsymbole** stehen für Begriffe, die durch **EBNF-Regeln** erklärt werden.
- **Terminalsymbole** (geschrieben in Anführungszeichen) sind – wie beim Syntaxdiagramm – die Grundelemente der formalen Sprache.

Jede EBNF-Regel hat die Form

A = Ausdruck

wobei A eine Nichtterminalsymbol ist und ein Ausdruck gebildet wird aus:

- Terminal- und Nichterminalsymbolen
- Operatoren zum Bilden von Ausdrücken

Auswahl	E1 | E2	für "E1oder E2"
sequentielle Komposition:	E1 E2	(direkt hintereinander schreiben) für „E2 folgt direkt auf E1“
Option:	[E1]	für „E1 kann 1- mal oder 0-mal (d.h. nicht) vorkommen“.
Wiederholung	{E}	für „E kann 0-mal oder mehrmals hintereinander vorkommen“.

- Eine **EBNF-Grammatik** ist eine Menge von EBNF-Regeln zusammen mit einem Startsymbol (Nichtterminalsymbol für den betrachteten Begriff, z.B. Satz).
- Jede solche **Grammatik G** definiert eine Menge von Wörtern, die als Sprache von G, geschrieben L(G), bezeichnet wird. (L steht für **L**anguage).

Ein Wort ist immer eine Folge von Terminalzeichen.

Man erhält jedes Wort w von L(G) durch eine Ableitung der Form $S \rightarrow E_1 \rightarrow E_2 \rightarrow ... \rightarrow E_k$

- wobei S das Startsymbol von G
- E_{i+1} entsteht aus E_i, dadurch daß ein oder mehrere Nichtterminale aus E_i durch die rechten Seiten ihrer Def. ersetzt werden oder Operatoren ausgewertet werden.
- $E_k \equiv w$

  Hier bedeutet "≡" die syntaktische Gleichheit.

  Bei dem abgeleitetem Wort w verzichtet man im allgemeinen auf die Anführungszeichen.

Bemerkung:

Die Java-Spezifikation verwendet eine etwas andere Schreibweise:

- *Nichtterminalsymbole* *kursiv*
  (z.B. *Ziffer* statt Ziffer)
- `Terminalsymbole` Programmiersprachenfont
  (z.B. `if` statt "if")

- Statt A = E schreibt man A: E
- Die Auswahl wird weggelassen und durch neue Zeile ersetzt.

  z.B. *Vorzeichen*: +
  -
- Die Option wird durch tiefgestelltes opt gekennzeichnet.

  z.B. E_{opt} statt [E]

Beispiel:

Zahl: Ziffer
Zahl Ziffer

Bemerkung:

Man kann {E} in EBNF in folgendermaßen definieren:

{E} = [E]| .
E {E}

Äquivalent dazu ist die Definition:

{E} = [E] |
{E}E

Beispiel:

Ein **Bezeichner** besteht aus einer nichtleeren Folge von Buchstaben oder Ziffern, beginnend mit einem Buchstaben.

Bezeichner sind z.B. A, A2D2, Wirsing

Keine Bezeichner sind 007, 1B, F.D.P *(Punkte sind keine Buchstaben.)*

EBNF-Grammatik:

Buchstabe = "A" |
"B" |
...
"Z" |
"a" |
...
"z"

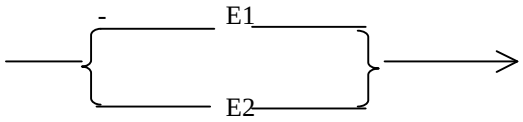
Buzi = Buchstabe |
Ziffer

Bezeichner = Buchstabe {Buzi}

2.3 Korrespondenz von Syntaxdiagrammen und BNF

Jeder EBNF-Operator läßt sich durch ein Syntaxdiagramm ausdrücken:

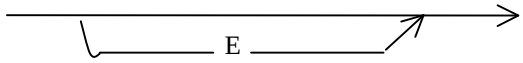
- Auswahl: E1 | E2 wird repräsentiert durch eine Verzweigung.



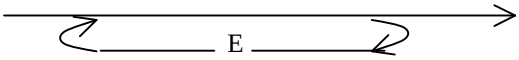
- Komposition: E1 E2 wird repräsentiert durch Hintereinanderfügen

E1→E2

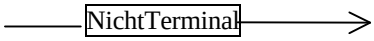
- Option: [E] wird repräsentiert durch



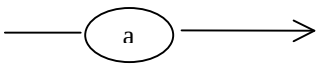
- Wiederholung: {E} wird repräsentiert durch



- Nichtterminalsymbol: Das Symbol NichtTerminal wird repräsentiert durch eine rechteckige Umrahmung:



- Terminalsymbol: Das Symbol „a“ wird repräsentiert durch eine ovale Umrahmung:



Umgekehrt läßt sich für jede durch Syntaxdiagramme definierte Sprache eine EBNF-Grammatik angeben. (Beweis ist nicht Teil dieser Vorlesung).